Class 6: R Functions

Zoe Matsunaga (PID:A16853288)

Table of contents

1. Function Basics 1
2. Generate DNA Sequences 2
3. Generate Protein Sequences 3

1. Function Basics

Let's start writing our first silly function to add some numbers:

Every R function has 3 things:

- name (we get to pick this)
- input arguments (there can be loads of these separated by a comma)
- the body (the R code that does the work)

```
add <- function(x, y=10, z=0){
  x+y+z
}
```

I can just use this function like any other function as long as R knows about it (i.e. run the code chunk)

```
add(1, 100)
```

```
[1] 101
```

```
add(x=c(1,2,3,4), y=100)
```

```
[1] 101 102 103 104
```

```
add(1)
```

```
[1] 11
```

Functions can have "required" input arguments and "optional" input arguments. The optional arguments are defined with an equals default value (`y=10`) in the function definition.

```
add(1, 100, 10)
```

```
[1] 111
```

> Q. Write a function to generate to return a DNA sequence of a user specified length? Call it `generate_dna()`

The `sample()` function can help here

```
#generate_dna <- function(size=5){}

students <- c("jeff", "jeremy", "peter")

sample(students, size = 5, replace = TRUE)
```

```
[1] "peter" "peter" "peter" "jeff"  "peter"
```

2. Generate DNA Sequences

Now work with `bases` rather than `students`

```
bases <- c("A", "C", "G", "T")
sample(bases, size=10, replace= TRUE)
```

```
 [1] "G" "T" "T" "C" "A" "C" "C" "C" "T" "A"
```

Now I have a working 'snippet' of code I can use this as the body of my first function version here:

```
generate_dna <- function (size=5) {
  bases <- c("A", "C", "G", "T")
  sample(bases, size=size, replace= TRUE) }
```

```
generate_dna()
```

```
[1] "T" "C" "T" "A" "A"
```

```
generate_dna()
```

```
[1] "C" "T" "T" "G" "C"
```

I want the ability to return a sequence like "AGTACCTG" i.e. a one element vector where the bases are all together.

```
generate_dna <- function(size=5, together = TRUE ) {
  bases <- c("A", "C", "G", "T")
  sequence<- sample(bases, size=size, replace= TRUE)

  if(together) {
  sequence <- paste(sequence, collapse= "")
  }
  return(sequence)
}
```

```
generate_dna()
```

```
[1] "AAGAA"
```

```
generate_dna(together=F)
```

```
[1] "G" "T" "G" "C" "C"
```

3. Generate Protein Sequences

> Q. Write a protein sequence generating function that will return sequences of a user specified length.

We can get the set of 20 natural amino-acids from the **bio3d** package (remember to 'install.packages("bio3d") if haven't already).

```
aa <- bio3d::aa.table$aa1[1:20]
```

```
generate_protein <- function(size=6, together = TRUE){

  ##Get the 20 amino acids as a vector
  aa <- bio3d::aa.table$aa1[1:20]
  sequence <- sample(aa, size=size, replace = TRUE)

  ## Optimally return a single element string
  if(together) {
  sequence <- paste(sequence, collapse= "")
  }
  return(sequence)
}
```

```
generate_protein(together =FALSE)
```

```
[1] "V" "F" "R" "Y" "A" "H"
```

Q. Generate a protein sequence of length of 6 to 12 amino acids.

```
#generate_protein (6:12)
```

We can fix this inability to generate multiple sequences by either editing and adding to the function body code(e.g. a for loop) or by using the R **apply** family of utility functions.

```
sapply(6:12, generate_protein)
```

```
[1] "GTKNYH"       "VNFTDHH"      "WIFRVEHS"     "MCCFCDWER"    "TQRYFSAVNT"
[6] "IKGNWSSKWMG"  "KGFQYVPSHWKT"
```

It would be cool and useful if I could get FASTA format output.

```
ans <- sapply(6:12, generate_protein)
ans
```

```
[1] "MCSEYE"       "VPGACLG"      "HSPYKTAK"     "AAPDVFFSF"    "VGRMNDCSIC"
[6] "NDFDWIARLSN"  "RWCHIWEWWHRW"
```

```
cat(ans, sep= "\n")
```

```
MCSEYE
VPGACLG
HSPYKTAK
AAPDVFFSF
VGRMNDCSIC
NDFDWIARLSN
RWCHIWEWWHRW
```

I want this to look like FASTA format with an ID line

```
> ID.6
FSTECG
> ID.7
TLMTHHC
> ID.8
TMNYWSPW
> ID.9
HFSHSGKAS
> ID.10
QWWFLSPRPQ
> ID.10
MKTTAWHFCIY
> ID.11
HFSNYIPCISDR
```

```
## Works, but too complicated!
cat(paste(">ID.", 6:12, "\n", sep="", ans), sep="\n")
```

```
>ID.6
MCSEYE
>ID.7
VPGACLG
>ID.8
HSPYKTAK
>ID.9
AAPDVFFSF
>ID.10
VGRMNDCSIC
```

```
>ID.11
NDFDWIARLSN
>ID.12
RWCHIWEWWHRW
```

```
id.line <- paste(">ID.", 6:12, sep="")
id.line
```

```
[1] ">ID.6"  ">ID.7"  ">ID.8"  ">ID.9"  ">ID.10" ">ID.11" ">ID.12"
```

```
##This works better!!
id.line <- paste(">ID.", 6:12, sep="")
seq.line <- paste(id.line, ans, sep="\n")
cat(seq.line, sep="\n", file="myseq.fa")
```

> Q. Determine if these sequences can be found in nature or are they unique? Why or why not?

After BLASTp searching my FASTA sequences against refseq_protein, length 6, 7, and 8 are not unique and can be found in the databases with 100% coverage and 100% identity.

Random sequences of lengths 9 and above are unique and can't be found in the databases.